Kimmo Koskenniemi

Notes on the Two-Level Morphology

Abstract

Some 24 years have passed since the advent of the two-level morphology. One motivation for the invention of the two-level model was computational efficiency. The power and capacity of present computers is now more than thousand times greater than what was available then. This article briefly relates the two-level model to the current state of the technology and identifies some possible areas where further work in the two-level framework would be worth while.

1. Background

Two-level morphology emerged in 1981–82 (Koskenniemi 1983) as a branch of the finite-state transducer cascade technology which Ronald Kaplan and Martin Kay had been developing since the 1970'ies (Kaplan and Kay 1994). Around 1980, the available memory of computers was about 1/10,000th of the memory of ordinary present computers, and the speed of computation was about 1/1000th of the speed of present computers.

In those days, the transducer technology was feasible for simple rule sets, but it was prohibitively heavy, if several transducers had to be composed together. The morphological (or graphemic) alternations of English could be handled then, but for Finnish, so many rules were needed that the results of the composition would have been too large. The obvious incentive for inventing the two-level model was to find a more tractable way to perform the required computations.

The two-level morphology turned out to be more than just as a computational trick or an optimized implementation of traditional phonological rewrite rules. It implied a different framework and a new interpretation for morphophonological alternations. Having fully parallel rules instead of the usual sequential cascade of rules had consequences such as:

1. There were no intermediate representations or results, i.e. only the morphophonemic and the surface forms existed.

2. The rule ordering did not matter. The rules could neither feed nor bleed other rules.

3. The rules were logical constraints defining the relation between the two representations rather than procedural actions converting strings to other strings.

4. Two-level rules could refer to conditions also on the surface, i.e. the final stage of the derivation (such as "ending up between vowels"), or even refer to relations between the morphophonemic and the surface (such as a "after a contracted diphthong"). The location of deletions and epenthesis can be controlled quite explicitly e.g. by referring to the resulting morphotactic structure.

These properties were partly good news implying certain rigidity and simplicity. Logical constraints are easy to understand and they have no opaque interactions. All rules must stick to the truth all the way. Therefore, one could not fix a fault in a rule by issuing another rule to patch it. The two-level rules did not tolerate exceptions. Some complicated rule conflicts and other interactions were also more problematic to handle in the two-level framework than in the cascading scheme.

The reader is advised to read the Karttunen (1991) or Karttunen and Beesley (2001) which elaborates the similarities and differences of the finite-state implementations of the cascaded transducer rules and two-level rules, and to the *Short history of the two-level morphology* by Karttunen (2001) for a more thorough survey.

The original idea was to restrict the use of two-level rules to alternations which were phonologically conditioned or which represented phonologically motivated changes. Everything else, i.e. irregular and purely morphological alternations, was supposed to be taken care of by the lexicon component. Complicated alternations can described using the mechanisms of a partitioned and linked lexicon; cf. the Xerox LEXC tool which is a systematic implementation of lexicons as finite-state transducers (Karttunen 1993).

In order to overcome some of the problems with the interactions between two-level rules, Lauri Karttunen devised a rule conflict resolution scheme which allowed one to write more than one two-level rule for same morphophonemes. The interactions that were resolved were cases where several rules give separate permissions for the occurrence of a pair, or where a rule acts as an exception to another rule, i.e. its context is a subset of the context of the second. This conflict resolution scheme allowed one to write more complicated two-level grammars. Thus, one could extend the domain of the rule component somewhat closer to the generative phonology. But, the heavy use of the conflict resolution mechanism resulted in a more complicated compilation and large transducers. (See below, in section 3, a note of the source of these conflicts.)

As the power and capacity of computers grew, more efficient and practical implementations cascaded rewriting systems such as the Xerox XFST, see e.g. (Beesley & Karttunen 2003). Full scale generative morphophonological descriptions and lexicons can now be handled in this way.

On the other hand, the rule formalisms for two-level morphology have been developed to some alternative directions. Partition based rule formalisms were proposed and their compilation was implemented by several researchers, including Alan Black, Graeme Ritchie, Stephen Pulman and Graham Russell (Black & al 1989), (Pulman & al. 1993). Edmund Grimley-Evans, George Kiraz and Stephen Pullum offer a different framework for implementing multi-tape versions of two-level rules and their compilation into finite-state transducers (Kiraz & Pullum 1996, Kiraz 2001). These have been applied for describing Semitic languages where interdigitation and circumfixing are serious challenges for all morphological analyzers. The partition based rules (with two and even with more than two tapes) retain the principle that no intermediate representations or results ever exist and that rules are strictly parallel. At least some of them are somewhat more restricted in describing repeated instances of morphophonemic correspondences.

2. Writing and debugging rule systems

One question that has occasionally been raised is whether writing two-level rules is easier or more difficult than designing rewrite rules. There appears to be no common measures of ease of writing rules and descriptions, except perhaps the amount of human effort required for a task. Unfortunately, we do not have even empirical data about the amount of work needed for designing morphological systems with different formalisms.

Building a two-level morphological parser consisting of a lexicon and a set of two-level rules may turn into a mess, unless certain planning and discipline is followed. The published monographs and articles do not fully cover the practical side cf. (Koskenniemi 1983 and 1991), so we give some guidelines here.

We assume here that the target language is in written form, has a norm and linguistic descriptions such as grammars which describe the inflection or other morphological and morphophonological phenomena.

1. Collect example words for each inflectional type, and for each morphophonemic alternation to be described with two-level rules. These examples are stored in a test file to which one will also put further examples of each morphophonemic alternation or phenomenon as needed to define the phenomena.

2. Identify morphemes tentatively within the examples. Add then some zeroes as necessary so that the morphs corresponding to the same morpheme are of equal length. Add zeroes to such positions that the morphs can be aligned with each other so that corresponding positions represent characters (or phonemes) which could reasonably alternate with each other.

3. The morphophonemic representations of each example are set up. If there is no variation among the morphs of a morpheme in a position, then the character itself can be used. Otherwise, a morphophoneme is postulated, cf. (Koskenniemi 1991).

Once we have prepared the test file, we can start designing the two-level rules. A rule for each morphophoneme is needed. If a morphophoneme has two alternative realizations, then one will be the default realization and a rule is needed for the other. We write a rule for the realization which has a linguistically cleaner and better motivated environment as a condition. Default realization of the morphophoneme usually represents the alternative which has no clean or systematic context which would account for its presence.

The value of the test file is twofold: first, it helps to design the morphophonemic representation and the rules, and secondly, it gives very accurate and pinpointed feedback on the possible shortcomings of the rules being developed. The compiler or the analyzing program gives exact character locations in the examples and names the rule or rules which fail, (cf. the TWOLC compiler by Lauri Karttunen at Xerox). The ease and success of the debugging depends much on how informative the feedback is. With the two-level compiler the designer of the rules gets quite detailed hints if something is wrong.

3. Automatic discovery of rules

The article which we referred to above (Koskenniemi 1991) was about sketching an informal discovery procedure for establishing the morphophonemic representations from paradigms of model words. That procedure was intended for humans who have at least some linguistic training and intuitions. It was hinted in the article that the mechanism could be automated, but nobody appears to have implemented the idea yet.

The underlying idea there was that one assumes a kind of distance metric among letters or phonemes as a computational formula. With such a measure, one can find more or less safely the positions for inserting the zeroes in the examples discussed above. The zeroes would be positioned so that the sum of the distances between corresponding characters of the same morph would be minimized. For example, if we have stems (and endings) such as *love*(*n*) and *lov*(*ia*), then one zero needs to be inserted to the latter stem. Suppose that we insert it in the middle, e.g. resulting in a stem *lo0v*, where the distances would be summed from pairs *l-l*, *o-o*, *v-0*, *e-v*. Apparently, the two last pairs have a great distance in our metric. Inserting the zero to the end would result in distances from pairs *l-l*, *o-o*, *v-v*, *e-0*. Here, the first three pairs certainly have a null distance, and the deletion of a vowel *e-0* has a fairly short distance. If we have a reasonably good approximation for the distance metric, we can expect that this procedure will converge into good and feasible insertions of the zeroes.

Once the zeroes are in place, the morphophonemic representations become fixed, if we simply define each alternation pattern of letters to be a morphophoneme of its own, e.g. *i-e-0*, *t-d*, *t-0* etc. We can use the sequence of alternatives as the name of the morphophoneme. This step of naming and determining the morphophonemic representations is quite deterministic and it poses no computational problems. In this approach we do not minimize the number of morphophonemes. If there is an alternation among different characters, it results in a different morphophoneme—even if a trained linguist would prefer to see the same underlying sound on the morphophonemic level.

The discovery of the two-level rules can utilize the same metric as was used for the positioning of the zeroes. One needs, however a model for rule types which are considered discoverable. Alternations triggered by the immediately following (or preceding) character are easy to discover and describe with rules. One simply collects the immediately following (or preceding) characters for each possible surface character of a morphophoneme, e.g. one set for *e-e* and another set for *e-0*. One computes the within set distances of the context characters for each realization. Good candidates for rules are those context sets where the within set distances are small, i.e. there is a generalization of the contexts (e.g. as "vowel" or "voiced consonant").

Similarly, one can program other rule types which are possible or common in the languages of the world. For vowel harmony rules, one can define general patterns according to vowel qualities (such as rounded/ unrounded, front/back, height etc.). For harmonies, one is interested in sets of vowels in the whole word-form rather than only in immediately neighbouring characters.

An important observation here is that many of the complications of rule interactions appear to disappear as we let the procedure to establish a larger number of morphophonemes than what linguists typically postulate. The left arrow rule conflict with fully included contexts vanishes because the general case and the more restricted case deal with different morphophonemes. The need to use multiple contexts or the need to resolve right arrow rule conflicts typically arises in occurrences in different positions. It probably is a good policy in the procedure to establish separate morphophonemes even for similar alternation patterns if they occur in morphotactically different positions. Our morphophonemic representation is now not really a generalization based on linguist's intuitions but rather just an account of actual alternations.

What makes the two-level framework here more interesting and promising than the traditional rewrite rules, is its smaller dimensionality. There are dramatically fewer representations to choose from because the insertion of zeroes already determines the rest of the process. Discovering the insertion itself only implies a small amount of enumerating. When the zeroes are in place, the establishing of morphophonemic representation is direct, and each rule can be discovered in isolation, and without interactions with other rules. This allows for repeating the process for several candidate positions of the zeroes, not only for the one which appeared to produce the minimal distances for the insertions. Thus, it is possible to search for a global optimum where both the positioning of the zeroes and the contexts of the two-level rules are taken into account with appropriate weights.

If one compares the above method with the discovery of cascaded rewrite rules, one notes the difference. With cascaded rules, one has to establish the underlying representation, the rules, and the sequencing of the rules, and these all affect each other. There is a distinct representation between each application of rules. If there are plenty of rules, the computation is likely to become difficult. (The discovery may be computationally heavy and/or sensitive to the input samples given to the procedure.)

4. Application to language learning

Computer assisted language learning (CALL) could benefit from the special property of two-level grammars, namely from the independence of each rule from the other rules. Let us suppose that a student is prompted to produce a certain inflected word-form involving several distinct morphophonological rules. A student of Finnish might try to produce *parroista* (‘from beards’ where the nominative singular is *parta*) and end up in producing *parraista*. Using traditional programs, the student would probably only get a feedback which indicates that the answer was not correct. But, within the two-level framework, a program can compare the reply and the morphophonemic representation of the correct target form using the same method as the linguist uses when testing rule sets. In our example, the representations to be compared would be:

```
p a r r~t a~o i~j s t a~ä
p a r r   a   i   s t a
```

The program could silently notice that the consonant gradation went fine, the vowel harmony was correct, the plural *i* was ok, but the stem final vowel alternation still needs training. Thus the feedback to the student could be mostly encouraging and the corrective actions would be directed to the essential areas.

One can claim that this kind of selective checking the correctness of alternations is simpler in the two-level framework than in the cascaded rewriting grammar. At least the mechanism needed is already known (similar to *pair-test* in Xerox TWOLC).

5. Summary

Currently, there are two common ways to describe and handle morphophonemic alternations in morphological analysis: the two-level model and the cascaded rewrite rule model. Both are actively used in current morphological parsers of commercial or academic origin. Probably a significant part of the present day spellers, inflecting synonym dictionaries and similar products make use of the two-level rules—if not the majority of those used in this part of the world. Some commercial parties, such as Xerox and ATT Research have opted for the rewrite rule model whereas some others use the two-level framework.

When the two-level model was first introduced, it was thought to suit for describing phonologically motivated, synchronic and at least partly natural phenomena rather than to accounting for more remote relations depending on long historical developments. Improvements in the two-level formalism have not changed this setup. Maybe one can still think that the two-level rules capture a part of the idea of phonological rules as some separable components of our linguistics competence.

A fully composed morphological analyzer as a finite-state transducer is independent of which of the two rule formalisms have been used. The two-level model can be efficiently interpreted even without this composition. The run-time code for that task is small enough to fit into a single sheet of paper.

Presently, the software supporting the rewrite model is quite proprietary even if it is available for purely academic applications, but also full scale two-level compilers have restrictions for their free use. It would be very desirable that truly free alternatives of both types would become available.

References

Beesley, Kenneth R. & Lauri Karttunen (2003) *Finite State Morphology*. Stanford, CA: CSLI Publications, Stanford University.

Black, Alan, Ritchie Graeme, Stephen G. Pulman& G. J. Russell (1987) Formalisms for morphographemic description. In *Proceedings of the 3rd Meeting of the European Chapter of the Association for Computational Linguistics*, (Copenhagen), pp.11–18.

Grimley-Evans, Edmund, George Anton Kiraz & Stephen G. Pulman (1996) Compiling a partition-based two-level formalism. In *COLING 1996: The 16th International Conference on Computational Linguistics*, Volume 1: 454–459.

Kaplan, Ronald M. & Martin Kay (1994) Regular Models of Phonological Rule Systems. *Computational Linguistics* 20.3: 331–378.

Karttunen, Lauri (1991). Finite-State Constraints. In *The Proceedings of the International Conference on Current Issues in Computational Linguistics. June 10-14, 1991*. Universiti Sains Malaysia, Penang, Malaysia.

—— (1993) *Finite-State Lexicon Compiler*. Technical Report ISTL-NLTT-1993-04-02. Palo Alto, CA: Xerox PARC.

Karttunen, Lauri & Kenneth R. Beesley. (2001) A short history of two-level morphology. A paper presented at *ESSLLI 2001*. URL: http://www.helsinki.fi/esslli/evening/20years/twol-history.pdf

Karttunen, Lauri, Ronald M. Kaplan & Annie Zaenen (1992) Two-Level Morphology with Composition. In *COLING 1992: The 15th International Conference on Computational Linguistics*, Volume 1, pp. 141–148.

Karttunen, Lauri, Kimmo Koskenniemi & Ronald M. Kaplan (1987) A compiler for two-level phonological rules. In Mary Dalrymple, Ronald Kaplan, Lauri Karttunen, Kimmo Koskenniemi, Sami Shaio & Michael Wescoat (eds.) *Tools for Morphological Analysis*. Report CSLI-87-108. Stanford, CA: Center for the Study of Language and Information, Stanford University.

Kiraz, Georg Anton (2001) *Computational Nonlinear Morphology with Emphasis on Semitic Languages*. Studies in Natural Language Processing. Cambridge: Cambridge University Press.

Koskenniemi, Kimmo (1983) *Two-level Morphology: A computational model for word-form recognition and production*. Publications of the Department of General Linguistics, University of Helsinki 11. Helsinki: University of Helsinki.

—— (1991) A discovery procedure for two-level phonology. In Laura Cignoni & Carol Peters (eds.) *Computational Lexicology and Lexicography. A Special Issue Dedicated to Bernard Quemada*, Volume 1, pp. 451–465. Linguistica computatzionale VI. Pisa: Giardini Editori e stampatori.

Pulman, Stephen G. & Mark R. Hepple (1993) A feature-based formalism for two-level phonology: a description and implementation. *Computer Speech and Language* 7.4: 333–358.

Contact information:

Kimmo Koskenniemi
Department of General Linguistics
P.O. Box 9 (Siltavuorenpenger 20 A)
FI-00014 University of Helsinki
kimmo(dot)koskenniemi(at)helsinki(dot)fi
www.ling.helsinki.fi/~koskenni